

RADC-TR-90-394
In-House Report
December 1990

AD-A232 410



2

ADDING A DECISION AID TO THE LACE/KB-BATMAN TESTBED

James H. Lawton II

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

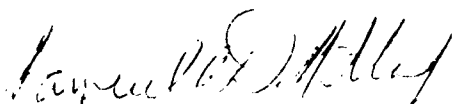
DTIC
ELECTE
MAR 11 1991
S B D

Rome Air Development Center
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700

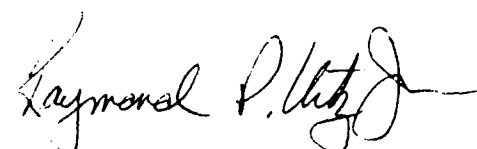
91 3 05 081

This report has been reviewed by the RADC Public Affairs Division (PA) and is releasable to the National Technical Information Services (NTIS) At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-90-394 has been reviewed and is approved for publication.

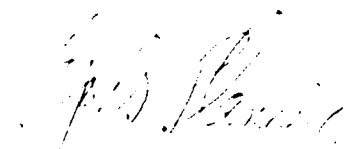
APPROVED: 

SAMUEL A. DINITTO, JR.
Chief, C² Technology Division
Directorate of Command & Control

APPROVED: 

RAYMOND P. URTZ, JR.
Technical Director
Directorate of Command & Control

FOR THE COMMANDER:



IGOR G. PLONISCH
Directorate of Plans & Programs

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (COES) Griffiss AFB NY 13441-5700. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave Blank)		2. REPORT DATE December 1990		3. REPORT TYPE AND DATES COVERED In-House Aug 88 - Nov 89	
4. TITLE AND SUBTITLE ADDING A DECISION AID TO THE LACE/KB-BATMAN TESTBED				5. FUNDING NUMBERS PE - 62702F PR - 5581 TA - 27 WU - 49	
6. AUTHOR(S) James H. Lawton II					
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Rome Air Development Center (COES) Griffiss AFB NY 13441-5700				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Rome Air Development Center (COES) Griffiss AFB NY 13441-5700				10. SPONSORING/MONITORING AGENCY REPORT NUMBER RADC-TR-90-394	
11. SUPPLEMENTARY NOTES RADC Project Engineer: James H. Lawton II/COES/(315) 330-3564					
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.				12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) The LACE/KB-BATMAN Testbed is an Object-Oriented decision aid testing environment and simulation. Its purpose is to explore and evaluate cooperating knowledge-based decision aids in a highly flexible environment. This report describes the addition of an existing decision aid, an air route planner, to the Testbed, and the issues involved in such an effort.					
14. SUBJECT TERMS Decision Aids, Route Planning, Computer Simulation, Knowledge-Based Simulation, Object-Oriented Programming				15. NUMBER OF PAGES 32	
				16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL		

Preface

Intelligent decision aiding technology is currently a widely studied topic in the Artificial Intelligence community. The Rome Air Development Center (RADC) is involved in this area of research in a number of different ways. This paper describes one of these ways: the addition of an existing decision aid, an air route planning system, to a prototype Testbed developed for the RADC. This testbed, called the Knowledge-Based Battle Management (KB-BATMAN) Testbed, was developed by MITRE (Washington) [Anken89].

This is not a position paper on how route planners, simulations, decision aids, or expert system testbeds should be implemented or combined. It is a report of completed and on-going research performed in-house at the RADC. Its purpose is to document the accomplishments and findings, both positive and negative, of adding a decision aid to our existing Testbed configuration.



Accession For	
NTIS GRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

1.0 - Introduction.

1.1 - Background.

Since 1965, the Rome Air Development Center (RADC) has been exploring ways to improve battle management simulation technology. In particular, we have concentrated on performing basic research in object-oriented simulation technology and in building a flexible environment for simulation development. The primary goal of this research has been to develop a simulation environment for use by the Air Force in support of Command, Control, and Communication (C³) research [Anken89]. As a result, an object-oriented, tactical battlefield simulation environment, named Land-Air Combat in ERIC (LACE), was developed.

LACE was extended to support the Knowledge-Based Battle Management (KB-BATMAN) Testbed Project, a contractual effort which investigated and designed a testbed environment for integrating and testing battle management decision aids. The KB-BATMAN effort was divided into two parts. First, a framework was to be constructed in which knowledge-based systems could be executed cooperatively. Second, the framework developed in the first phase was to be used to integrate a group of knowledge-based systems. LACE was extended to simulate the resulting plans these system cooperatively developed [Anken89, Nugent88].

Figure 1 shows an example KB-BATMAN framework [Grimshaw89]. This prototype framework initially had two cooperating expert systems. The Mission Planner, AMPS (A Meta-Level Planning System), was developed to operate at the Tactical Air Control Center (TACC) level and would be used to generate Air Tasking Orders (ATOs). The Intel Aid, TAC-OB (Tactical Order of Battle), is an intelligence aid that generates a prioritized target list. It is also known as INTEL: Intelligence Analysis System.

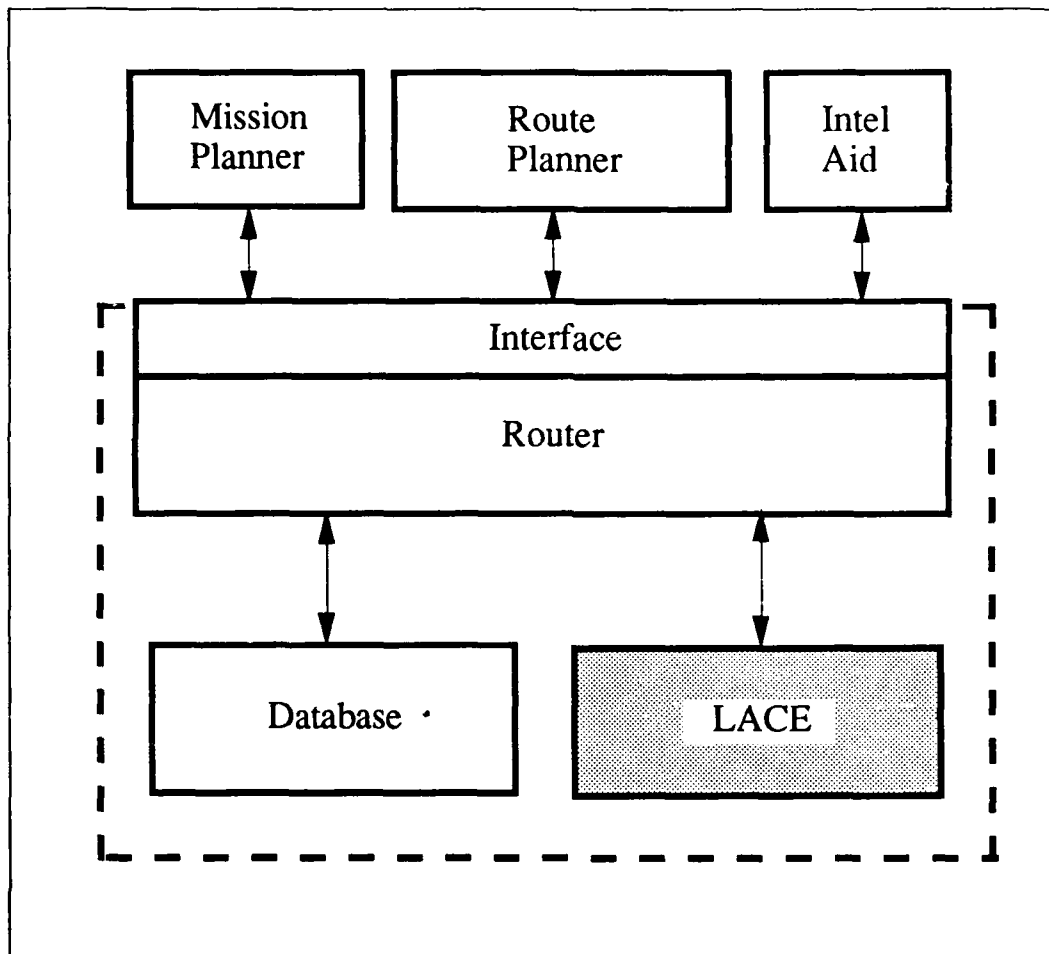


Figure 1. KB-BATMAN Testbed with Battle Management Decision Aids

A typical use of this scenario might proceed as follows: TAC-OB produces a prioritized list of targets based upon target information in the Database and knowledge from intelligence specialists. AMPS then uses this list to produce ATOs for strike missions against the targets. LACE then simulates flying these missions against the defended targets, reporting the results to the database. Based upon these results, the decision aids would be used to develop new missions, and the whole process could be cycled through again. The Testbed (the part of Figure 1 enclosed in dashed lines) is responsible for controlling the communication between and maintaining a common database for these systems [Anken89, Grimshaw88]. The Route Planner in figure 1 is the decision aid that

was added and is the topic of this paper. For the remainder of this paper, the LACE environment and the KB-BATMAN Testbed environment will be collectively referred to as the Testbed environment.

1.2 - The Problem.

In the prototype Testbed environment, AMPS plans missions to only a limited level of detail. An air facility for take-off is specified, and, of course, the target, but there is no information on how a strike mission should fly to its target. The most obvious expert system to add to the environment was a mission route planner. A system of this nature could analyze the threat picture of the battle area and plan a flight route that gave the mission a good chance of arriving at its target, performing its task, and making it back home.

Our problem then was to determine the feasibility of adding a decision aid, namely a mission route planner, to the Testbed environment. Note that the point was not to just add a route planner to the Testbed environment, but rather to determine through experimentation whether adding existing decision aids to it was feasible, and to pinpoint the issues, such as selection criteria, and potential problems associated with such integration.

1.3 - The Approach.

In order to solve this problem, an experiment was set up. An attempt was made to find an existing route planning expert system which would require little modification to add to our Testbed environment. When found, the code and documentation would be examined to determine if the system's implementation would be feasible to modify in-house and use in the Testbed. As a last resort, a simple planner would have been written in-house to add to the environment.

However this would have indicated that our Testbed was not providing a very flexible testing environment.

2.0 - Procedure.

2.1 - The Search for Planners.

The first effort of this experiment was to gather as much information on existing mission route planners as was possible. This was done in three ways: A literature search through the RADC technical library; discussions with engineers in other sections within the RADC who were known to be doing or sponsoring work in the area of intelligent route planners; and by placing a request on the Internet Artificial Intelligence bulletin boards for information on Lisp-based route planning systems. A partial list of the responses from the various searches is given in Appendix A.

2.2 - Choosing a Planner.

We were looking for a planner that met the following (minimal) criteria: (1) It was to be written in Lisp, preferably running on a Symbolics Lisp Machine. (2) It needed to be designed such that the planning code could be easily extracted and interfaced to. (3) It needed to be fast, on the order of planning single strike mission route in less than five minutes in the absolute worst case.

The Lisp language was required because the Testbed is written in Lisp, running on the Symbolics platform. The "extractability" requirement was important primarily from a programmer's point of view. Even with documentation, a system in which the interface code is heavily entangled with the performance code is often difficult (at best) to interface to, and even harder to modify. The speed of the planner was important to maintain the focus of the

experiment: to study adding decision aids to the Testbed, not to get a perfect planner.

After reviewing the many papers and system descriptions we received, two systems were chosen for further study: The Personalized Route Planner (PRP) and the Route Planner Development Workstation (RPDW). The following sections give an account of what was done with each system.

2.2.1 - The Personalized Route Planner.

The Personalized Route Planner (PRP) was built by Decision Sciences Consortium, Inc., to demonstrate the concepts and techniques of adaptive decision aiding. This is an approach to designing decision aids that attempts to make the aid adapt to different users according to their level of skill, experience, and their personal preferences [Mullin87].

The PRP was chosen because it was written in Lisp running on a Symbolics Lisp Machine, and because it was claimed that the planning and interface code were separated. We knew that the system did not meet our speed requirements, but we hoped that the planning code running without the graphics interface would be faster. A copy of the code for the PRP was obtained from Mr. Robert Kruchten, who works in the Decision Aids branch of the RADC. The code was analyzed and the specific parts required to make a usable route planner, without the user interface, were extracted.

The PRP generates routes by starting with a simple input route between the start and the goal, which is usually just a straight line. This initial route is then improved over a series of iterations. At any given time, there is a current "best" route. The user may stop the system and accept the current route, or allow it to continue [Mullin87].

The code used to analyze and improve routes was extracted from the PRP system. This was not as simple as it sounds. The PRP was purported to be written using three software modules: one for threat analysis, one for route evaluation, and one for route generation [Mullin87]. The actual code that performed these functions, however, was spread out over different files. It was written with user interface code deeply intertwined with the planning code, making the extraction of the algorithm specific code very difficult. This may partly be explained by noting that in the contract for the PRP, it was made clear that there was no requirement for the PRP system to interface with other systems [Mullin87].

In order to make the PRP work in the Testbed environment some additional code was necessary. The PRP uses information about surface-to-air missile (SAM) sites to generate improved routes. Code was written in-house to translate the way the Testbed environment represented SAM-sites to the way the PRP did.

Once this interfacing was completed, the PRP was tested by asking it to improve a straight line route from a friendly air base in the Testbed environment to an enemy one. The results were rather unexciting. Even after letting the system run for more than 10 hours, the "improved" route was still just a straight line.

This was rather disheartening. Before attempting to modify the PRP, the RADC personnel were given a demonstration of the system by personnel from Decision Sciences Consortium, Inc. The demonstration went well, showing a series of viable routes that the PRP had planned earlier that day. They claimed that a reasonably good route could be found in forty minutes or less. Although this was extremely slow for our purposes, we decided to analyze it anyway, with the hope that these claims were exaggerated. (They were, of course, but not in the way we hoped).

Because of these problems with speed and reliability, it was decided that the PRP was not a feasible alternative for a route planner in the Testbed environment.

2.2.2 - The Route Planner Development Workstation.

The second route planner examined for use in the Testbed environment was extracted from the Route Planner Development Workstation (RPDW), a system developed at the Jet Propulsion Laboratory, California Institute of Technology, in Pasadena, California. The RPDW was designed to facilitate the development and evaluation of route planning algorithms. The system provided a number of basic elements needed to study route planner algorithms, including a basic cartographic substrate, functions to transform the cartographic data into forms usable by various route planning algorithms, and some evaluation metrics and means to report test results. Various planning algorithms could be tested without any re-coding of the basic environment [Cameron85].

The RPDW was chosen because it was written in Lisp running on a Symbolics Lisp Machine, because the planning algorithms were all separate from the more basic testing elements (described above), and because there were several planning algorithms included in the system. The algorithms varied in complexity, thus increasing the chances that at least one could meet our timing requirements.

A copy of the RPDW was obtained from Mr. Matt Swanson, Topographic Laboratory, US Army, Fort Belvoir, Virginia. This code was also analyzed, and the parts used just for route planning, without the user interface, were extracted. The extraction of the planning algorithms and necessary support code from the RPDW was rather simple. The system was written in a very modular fashion. The planning code had very few calls directly to the user interface. These were easily edited out.

There are two primary aspects of the RPDW planning system: a "surface" and a planning algorithm.

2.2.2.1 - The Surface.

The "surface" is a structure that describes the threat picture of a given geographic area (in our case, an area in central and eastern Europe). The primary part of this structure is a two dimensional array of integers. In our implementation, each element of this array corresponds to a 10 kilometer by 10 kilometer region (or "block") in central Europe. The integer in the array element is the "cost" of being in that block. For purposes of simplicity, the "cost" of a block in our implementation is the number of SAM-sites that could fire at an aircraft flying through that block.

A second major aspect of the surface is a transition array. This is three dimensional array that describes the cost of moving from one block in the cost array to an adjacent block. If the dimensions of the cost array are N by M, then the transition array is dimensioned N by M by 8. This captures the cost for moving in any of eight directions from any given spot in the cost array. For simplicity, the transition costs were computed as follows: if the movement was from a higher cost block to a lower cost one, the transition cost was -1; if the movement was from a lower cost block to one with a higher cost, the transition cost was +1; moving between equal cost blocks had a transition cost of 0.

The RPDW had the basic "surface" code included. This was extracted, and modified slightly to use with the Testbed environment. The only additional code needed to be written in-house for this aspect of the RPDW was the functions to generate the cost and transition arrays, and that to create instances of a surface.

2.2.2.1 - The Planning Algorithms.

The RPDW also included several algorithms which were designed to find least cost paths through a given surface. These needed little modification to run with the Testbed's map surface. The only additional code that was needed for this aspect of the RPDW was to make instances of the given route planning algorithms.

All applicable algorithms included were tested by asking a planner to generate a route plan from a friendly air base to an enemy one. Some took hours to find a route, others only seconds. Since the point of the experiment was to find a route planner that could be added to the Testbed environment, not *the* best route planner, the algorithm that worked the fastest and produced the "smoothest" routes in the tests was chosen. This was an algorithm written by Marc G. Slack of the Jet Propulsion Laboratory, Pasadena, California. Appendix B gives a more detailed description of the tests that were performed.

3.0 - Conclusions.

The results of this study strongly support the claim that existing decision aids can be added to an environment like the LACE and KB-BATMAN Testbed. This section identifies some of the basic criteria a decision aid must meet in order to integrate it into a cooperating environment of this type.

First, the decision aid must be applicable to the domain of the Testbed. This may not be as obvious as it sounds. For example, a number of expert systems that determined routes for ground vehicles were offered in response to our request for route planners. While the Testbed may be able to support such a system, there is no need for one now.

Second, the Testbed environment must be able to provide the type and quantity of information required by the candidate decision aid. Several responses

were received offering expert systems that did plan air routes, but relied upon information that was not available in our environment, such as elevation data and weather conditions, or did not use a compatible coordinate system. Although the systems that we could not use for this reason help to point out the deficiencies of our Testbed, one must keep in mind that our Testbed is still rather young in its development, and that no system can possibly contain knowledge about every possible domain.

The next criterion (another seemingly obvious point), is that the code be modular in nature. Although decision aids that use the object-oriented paradigm have the best chance of meeting this criterion, it is not required. What is necessary is that the interface to the expert system be structured such that there are a few simple queries made to the system through software which return some desired information. For example, the route planner is given a starting point and a goal point, and returns a path (a list of waypoints) for a mission to follow to its target. The details of how it computes this path are not important, only that it returns a reasonable path.

The fourth criterion, suggested by the above, is that the user interface of the expert system should be separate from the "working" code. For example, the code that actually does the planning for the route planner was easily extractable from that which handles the graphics and input/output. This is necessary to be able to develop a software interface to the system.

Finally, a detailed understanding of the inner workings of the decision aid should *not* be necessary. An understanding of the functionality of the system, in order to develop the interface between the Testbed and the expert system, should be sufficient. This understanding should be found by reading the documentation, or at worst by scanning through the code itself. (One would have to thoroughly examine the RPDW documentation and the program code to find out exactly how

Mr. Slacks algorithm works. This, I believe is a plus to the experiment, however. The algorithm worked on the simple surface that was created for the Testbed environment without any "tweaking" at all.)

Appendix A

The following is a listing of a number of the references and systems that were received in the search for route planners. It is not even close to a complete list; that would be a paper in itself. Rather, it is a list of the ones that appeared directly relevant to the experiment.

Papers

Bahnij, Robert B., Major, and Stephen E. Cross, Major. "A Fighter Pilot's Intelligent Aide for Tactical Mission Planning", Artificial Intelligence Laboratory, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1986.

Bradshaw, Jeffery S., 2Lt. "A Pilot's Planning Aid for Route Selection and Threat Analysis in a Tactical Environment," Master's Thesis, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, DTIC number: AD-A178 871, 1986.

Broadwell, M. M., and D. M. Smith. "Interfacing Symbolic Processes to a Flight Simulator," 1986 Summer Computer Simulation Conference Proceedings, 1986.

Chen, D. C. "An Expert Planner for the Dynamic Flight Environment," Proceedings of the National Aerospace and Electronics Conference (NAECON), 1985.

Coleman, Jr., James P. "Penetration Analysis Aids for the TAF Mission Planning System," Georgia Tech Research Institute, Presented in the Conference on Applied Military Decision Aids & Support Systems, Fort Wayne Indiana, September 16-17, 1986.

"Experiments with a Knowledge-Based System on a multiprocessor," Knowledge Systems Laboratory Report KSL 87-61, Computer Science Dept., Stanford University, 1988.

Kruchten, Robert J. "Decision Aid for Threat Penetration Analysis," Advanced Computer Aids in the Planning and Execution of Air Warfare and Ground Strike Operations: Conference Proceedings, 1986.

Lizza, Carl S., Capt, and Lt Gretchen Lizza. "Path-Finder: An Heuristic Approach to Aircraft Routing," Air Force Wright Aeronautics Laboratory, Wright-Patterson AFB, Ohio, NAECON 85.

Mulvey, John M., and Stravos A. Zenios. "Real-Time Operational Planning for the U.S. Air Traffic System," Elsevier Science Publishers B.V., (North-Holland), 1987.

Urlings, P.J.M and A.L. Spijkervet. "Expert Systems for Decision Support in Military Aircraft Mission Preparation," National Aerospace Lab, Amsterdam (Netherlands), 27 pp., Rpt No. NLR-MP-87013-U Available from NTIS as PB88-166145/WCC, 10 Feb 87.

Systems

Cameron, J., et al. *Route Planner Development Workstation*, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California, 1985.

Mullin, T.M. and J.R. McIntyre. *Personalized Route Planner*, Decision Science Consortium, Inc., Falls Church, Virginia, 1988.

Appendix B

This section contains the results of testing the route planners examined during the experiment.

I. The Personalized Route Planner.

[Information for this section was taken from Mullin87 and the Personalized Route Planner program code.]

The Personalized Route Planner (PRP) plans flight paths through threats in a local cartesian coordinate system. The system is written in Lisp for a Symbolics Lisp machine, using the Flavors object-oriented programming system. Each threat is an instance of a particular surface-to-air missile class (flavor), with a given set of coordinates, lethality, range, and other relevant instance variables. The planner starts with an initial route from a starting point in the local coordinate system to a goal point. This initial route is usually just a straight line from the start to the goal, although it can be divided into segments with waypoints. The planner then attempts to "improve" this initial route over numerous iterations of splitting up segments and bypassing the threats. Thus, at any given time, there is a current "best" route. If time is a constraint, as it always is in a planning domain, one can let the PRP run for some maximum time, and take the current best route, or let it run until it has exhausted its search.

For this experiment, PRP type threat instances were made from the LACE threat instances, using the LACE Map Display System's coordinates for the local coordinate system. Instances of initial routes were made; they were simply straight lines (i.e., just start and goal points) from points in the friendly area to points in the enemy area through a number of overlapping threats. The planner

was then asked to "improve" these routes (one at a time, of course). It was allowed to run for more than ten hours for each. The result was always still just a straight line, with one or two waypoints along the route.

It is possible that the modifications that were made to the PRP to adapt it to the LACE environment were not enough to make it work properly. Nonetheless, from these initial tests, it was decided that the PRP was not a feasible system for our purposes.

II. The Route Planner Development Workstation.

[Information for this section was taken from Cameron85 and the Route Planner Development Workstation program code.]

The Route Planner Development Workstation (RPDW) contained a number of route planning algorithms. The results for the ones tested are shown below. The RPDW was written in Lisp for a Symbolics Lisp machine, using the Flavors object-oriented programming system.

Each algorithm included in the RPDW was designed to plan a least cost path through a surface.". A surface is a structure (flavor instance, actually) that describes a particular geographic region. The primary parts of a surface are a cost array and a transition array. A cost array is simply a two dimensional array of numbers, where each number corresponds to some geographical region on a map. The number represents the "cost" of being over that particular geographical area. A transition array is a three dimensional array that corresponds to the same geographical region as the cost array in the first two dimensions (row and column), but has a third dimension whose elements represent the cost from moving from a given location (row, column) to an adjacent location. Thus total cost for moving from one place in the (two dimensional) array

to an adjacent place is the sum of the "cost array" cost of the new place and the transition cost of moving from the current place to the new one.

Each algorithm is given a start coordinate and a goal coordinate (row and column pairs) and returns a list of points (also row and column pairs) that represents that algorithm's version of a least cost path from the start to the goal.

The threat array used is shown in Figure B-1. For the LACE environment, each number represents how many surface-to-air (SAM) sites could hit an aircraft flying in the corresponding geographic region.

The Algorithms.

Each planner was asked to plan three paths:

Path-1: Start = (5 22); Goal= (26 35)

Path-2: Start = (19 1), Goal= (27 28)

Path-3: Start = (13 7); Goal= (28 39)

The following is a brief description of each algorithm, along with the results of that algorithm finding paths for the above. Note that each algorithm was modified slightly so that it would not run longer than two minutes. If a particular planning algorithm could not return a complete path when two minutes was reached, it simply returned what it had done so far. Two minute times for some algorithms may have been a result of "infinitely" looping in a certain area.

Marc's Simple Route Planner - Written by Marc G. Slack, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA. Although it is "simple", it is undocumented.

Jon's A* Planner - Written by Jonathan M. Cameron, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA. It is a classic A* ("A star") planner.

Jonathan's Local Planner - Also written by Jonathan M. Cameron, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA. It is strictly a simple local planner, choosing the lowest "cost" immediate neighbor as the next move.

Swannies Semi Local Planner - Written by Matthew C. Swanson, U.S. Army Engineering Topographic Laboratories, Ft. Belvoir, VA. It attempts to minimize local affects of the cost surface by basing its "next node" selection criteria on a "look ahead" (i.e. depth search) basis.

Table 1, below, summarizes the results of the above algorithms. Figure B-1 is a depiction of the threat array used for testing these planners. Figures B-2 thru B-5 show the routes planned by each planner.

Planner	Time (seconds)			Completed?	Results
	Path-1	Path-2	Path-3		
Marc's Simple Route Planner	15	15	18	Yes	Figure B-2
Jon's A* Planner	4	4	5	Yes	Figure B-3
Jonathan's Local Planner	121	1	1	No	Figure B-4
Swannies Semi Local Planner	121	1	1	No	Figure B-5

Table B-1 - Planner Execution Results

Figure B-1
LACE Threat Array
[Note that "." represents a 0 threat]

The output from Mark Slack's Simple Planner. This is the one that was actually added to the LACE environment.

```

.....G
.....X
.....X
.....X
.....X
.....X
.....X
.....X
.....X
.....X
SXXXXXXXXXX.X
..XX.....X
...XX.....XX
.....XXXXXXX

```

A 20x20 dot grid with a path of 'X' marks starting from 'S' at (10, 1) and ending at 'G' at (18, 18). The path consists of 18 'X' marks. The 'S' is at (10, 1) and the 'G' is at (18, 18). The path starts at (10, 1), goes up to (10, 4), then right to (11, 4), then up to (11, 7), then right to (12, 7), then up to (12, 10), then right to (13, 10), then up to (13, 13), then right to (14, 13), then up to (14, 16), then right to (15, 16), then up to (15, 18), then right to (16, 18), then up to (17, 18), then right to (18, 18).

[illegible]

The output from Jonathan Cameron's A* planner.

Path-2
Time = 4 seconds

A dot plot showing the distribution of the number of letters in the names of 25 presidents. The x-axis represents the number of letters (10 to 18), and the y-axis represents the frequency (0 to 25). The distribution is roughly bell-shaped, peaking at 14 letters with a frequency of 8.

Number of Letters	Frequency
10	1
11	2
12	3
13	4
14	8
15	4
16	2
17	1

A 20x20 grid of dots. The following characters are placed on the grid:

- Row 1: Column 19 has 'G'.
- Row 2: Column 19 has 'X'.
- Row 3: Column 19 has 'X'.
- Row 4: Column 19 has 'X'.
- Row 5: Column 19 has 'X'.
- Row 6: Column 19 has 'X'.
- Row 7: Column 18 has 'X'.
- Row 8: Column 19 has 'X'.
- Row 9: Column 19 has 'X'.
- Row 10: Column 19 has 'X'.
- Row 11: Column 19 has 'X'.
- Row 12: Column 19 has 'X'.
- Row 13: Column 19 has 'X'.
- Row 14: Column 19 has 'X'.
- Row 15: Column 19 has 'X'.
- Row 16: Column 19 has 'X'.
- Row 17: Column 19 has 'X'.
- Row 18: Column 19 has 'X'.
- Row 19: Column 19 has 'X'.
- Row 20: Column 19 has 'X'.

The pattern of 'X's forms a shape that is roughly triangular, with the base at the bottom and the apex at the top right. The letters 'G', 'S', and 'C' are placed at the top right, bottom left, and bottom center of the grid respectively.

The output from Jonathan Cameron's simple local planner. Note that cycling apparently occurred in planning path-1.

[illegible]

```

.....G.....
.....X.....
.....XX.....
.....X.....
.....XX.....
.....X.....
.....X.....
...X.....X....
..XXX.X.....XXXX
SXXXXXXXXX...XXX..X
.....XX...X....
.....XXX.....

```

```

      .G.
     .X.
    .X.
   .X.
  .X.
 .X.
XXXXXXX
  X
 XXXX
   XX
    X
   XX.XX.X
  XX.XXX
   XX
  X
 .X
.S

```

The output from Matt Swanson's Semi-Local Planner. Note that cycling apparently occurred in planning path-1.

[illegible]

```

      C
      X
    XX
      X
    XX
      X
    X
  XXX X
XXXXXXX XXX X
      XX X
      XXX

```

```

      .G.
    .X.
  .X.
.X.
.X.
.X.
XXXXXXX
      X
    XXXX
  .XX.
.X.
XX. XX. X.
XX.   XXX
XX
.X.
.X.
S

```